QUIZ

True or False: You can traverse a stack through its nodes.

True

False

👏 Stacks can only view and make changes to the top node in the stack.

Complete `Stack`'s `is_empty()` method so that you can use it to check if a stack is empty:

```
class Stack:
 def __init__(self, limit=1000):
   self.size = 0
   self.top_item = None
   self.limit = limit

 def is_empty(self):
   ???
```

```
return self.size = 0
```

```
return size = 0
```

```
return self.size == 0
```

👏 You got it!

Which values go where in a stack's `__init__()` method?

```
def __init__(self, limit=55):
  self.size = #1
  self.top_item = #2
  self.limit = #3
```

#1: limit, #2: 0, #3: None

#1: 0, #2: limit, #3: None

#1: 0, #2: None, #3: limit

When you instantiate a stack, the `size` is 0, there is no `top_item` yet (so we set it to `None`), and we give `limit` the passed in or default integer value to help us limit the size of our stack to prevent stack overflow.

Show less

This stack class method could be useful because it allows us to:

```
def some_method(self):
  return self.limit > self.size
```

Check the number of nodes that exist in the stack.

Check if there's enough space to add another node.

If the limit is greater than the current size of the stack, we know that there is still room to add another node.

In `__init__()`, what purpose does the line `self.limit = limit` serve?

```
def __init__(self, limit=8000):
  self.size = 0
  self.top_item = None
  self.limit = limit
```

It sets a limit for the number of nodes that can be added to the stack **at any one time.**

To prevent stack overflow, it is necessary to limit the number of nodes that are added to the stack.

Fill in the method name:

```
def ???(self, value):
  if self.has_space():
    item = Node(value)
    item.set_next_node(self.top_item)
    self.top_item = item
    self.size += 1
  else:
    print("No more room!")
```

`is_empty`

`peek`

`push`

👏 `push()` checks to see if there is enough space in the stack to add a new item and adds the item if there is space.

Fill in the method name:

```
def ???(self):
  if self.size > 0:
    item_to_remove = self.top_item
    self.top_item = item_to_remove.get_next_node()
    self.size -= 1
    return item_to_remove.get_value()
  print("This stack is totally empty.")
```

`peek`

`is_empty`

`pop`

👏 `pop()` checks to see if the stack is empty and removes the top item from a stack if there is an item to remove.

Fill in the method name:

```
def ???(self):
  if self.size > 0:
    return self.top_item.get_value()
  print("Nothing to see here!")
```

- `pop`
- `push`
- `peek`

`peek()` returns the value of the top item in a stack.

- `is_empty`

Which method would this line of code belong in?

```
self.size -= 1
```

- `pop()`

This line makes sense in `pop()` where you are removing an item from the stack.

- `peek()`
- `__init__()`
- `push()`